The Troubling Aspects of a Building Block Hypothesis for Genetic Programming

Una-May O’Reilly
Franz Oppacher

SFI WORKING PAPER: 1994-02-001

SFI Working Papers contain accounts of scientific work of the author(s) and do not necessarily represent the views of the Santa Fe Institute. We accept papers intended for publication in peer-reviewed journals or proceedings volumes, but not papers that have already appeared in print. Except for papers by our external faculty, papers must be based on work done at SFI, inspired by an invited visit to or collaboration at SFI, or funded by an SFI grant.

©NOTICE: This working paper is included by permission of the contributing author(s) as a means to ensure timely distribution of the scholarly and technical work on a non-commercial basis. Copyright and all rights therein are maintained by the author(s). It is understood that all persons copying this information will adhere to the terms and constraints invoked by each author’s copyright. These works may be reposted only with the explicit permission of the copyright holder.

www.santafe.edu
The Troubling Aspects of a Building Block Hypothesis
for Genetic Programming

Una-May O’Reilly
Santa Fe Institute
1660 Old Pecos Trail, Suite A
Santa Fe, NM, 87505
unamay@santafe.edu

Franz Oppacher
School of Computer Science
Carleton University
Ottawa, Ont., CANADA, K1S 5B6
oppacher@scs.carleton.ca

Abstract
In this paper we rigorously formulate the Schema Theorem for Genetic Programming (GP). This involves defining a schema, schema order, and defining length and accounting for the variable length and the non-homologous nature of GP’s representation. The GP Schema Theorem and the related notion of a GP Building Block are used to construct a testable hypothetical account of how GP searches by hierarchically combining building blocks. Since building blocks need to have consistent above average fitness and compactness, and since the term in the GP Schema Theorem that expresses compactness is a random variable, the proposed account of GP search behavior is based on empirically questionable statistical assumptions. In particular, low variance in schema fitness is questionable because the performance of a schema depends in a highly sensitive manner on the context provided by the programs in which it is found. GP crossover is likely to change this context from one generation to the next which results in high variance in observed schema fitness. Low variance in compactness seems fortuitous rather than assured in GP because schema-containing programs change their sizes essentially at random.

Introduction
Based upon the Schema Theorem [Holland 1975], the Building Block Hypothesis [Holland’92, Goldberg’89] is the original account of the search power in Genetic Algorithms (GAs). Holland’s work is fundamental analysis, and from it more precise or clarified explanations (some diverging from a schema-based approach) of GA search behavior have been pursued. Some experimental [Mitchell, Forrest et al. 1991; Forrest and Mitchell 1992] and theoretical [Grefenstette and Baker 1989; Radcliffe 1991; Altenberg 1994] research has even diminished the value of the Building Block Hypothesis as a description of how GAs search or the source of the GA’s power. For example, it is unclear which classes of search functions (even those with building block structure) the GA will solve faster than other search techniques such as hill climbing [Mitchell, Holland et al. 1993].

The Schema Theorem and Building Block Hypothesis were but the start of the process of detailed analysis of GAs. Genetic Programming (GP) [Koza 1992] is “a (relatively) new kid on the evolution-based algorithms block”. The enthusiasm to apply GP has outpaced the attention paid to explaining it as a search technique. Koza [Koza 1992] presents but a brief sketchy analogy with the GA Schema Theorem and Building Block Hypothesis. Another approach based upon population genetics analysis [Altenberg 1994] has great potential however the assumptions of the simplified “generation 0” model raise the same unresolved crucial issues we shall present here. In this paper we go through the exercise of rigorously formulating the Schema Theorem for GP. To do so involves defining a schema, schema order, and defining length and accounting...
for the variable length and the non-homologous nature of GP’s representation. Once a GP Schema Theorem is formulated it is possible to define a building block. However, because this definition is imprecise (i.e. it includes a random variable) it is only possible to formulate a high level and preliminary description of GP search as a hierarchical building block combination process. That reality (i.e. actual GP search behavior) conforms to this description is a conjecture that is predicated upon some assumptions in GP which intuitively seem weak. In particular, it is unlikely that the properties a building block must exhibit exist in GP. A GP building block needs to have consistent (low variance), above average properties in fitness and compactness. Low variance in schema fitness is questionable because the performance of a schema is related in a highly sensitive manner to the context provided by the programs in which it is found. The GP crossover is likely to change the context of a schema from one generation to the next which likely results in high variance in observed schema fitness. Consistent compactness which is the expectation that a schema is unlikely to be disrupted, seems good fortune rather than assured in GP because the size change in GP programs which contain the schema is essentially random. Interestingly Altenberg essentially reaches the same impasse. His “generation 0” model of GP assumes that the constructional fitness value of a block of code (i.e. the factor by which a given block of code changes the fitness of a program to which it is added) has a relatively stationary distribution and that crossover proliferates blocks of code but never deletes them [Altenberg 1994].

Section 1 presents definitions pertaining to GP schemas. Section 2 is a GP version of the Schema Theorem. In Section 3 a Building Block is defined using the GP Schema Theorem. From this a hypothetical interpretation of GP search as a hierarchical building block combination process is presented. In Section 4 we discuss why the assumptions underlying the described process are questionable. We conclude by noting that as yet there is no detailed explanation of GP success or any theoretical conviction for the conclusion that GP is more powerful than non-population based optimization techniques. We suggest that explicit schema based function experimentation, statistical analysis of landscapes and comparison of GP with other optimization techniques offer potential insight into investigating GP search behavior.

Section 1: Schema Definition

Schemas, or similarity templates, are an arbitrary way of defining subsets. What is a schema in GP? According to Koza,

a schema in GP is the set of all individual trees from the population that contain, as subtrees, one or more specified trees. A schema is a set of LISP S-expressions (i.e., a set of rooted, point-labeled trees with ordered branches) sharing common features. [Koza 1992] (p 117-118)
A is a tree because all of the leaves which are variables or constants require no arguments.
B is a fragment not a tree because it has leaves (< and dec) which require arguments.

Figure 1 Schema Examples

The distinctive aspect of Koza's schema definition is that a schema is an S-Expression which is isomorphic to a tree. See Tree A of Figure 1. This implies that no schema is defined by a fragment or tree-like structure (as in Fragment B of Figure 1) where a primitive requiring arguments corresponds to a leaf (i.e. it is missing arguments).

In this respect we choose to be more general. Anticipating the implications of the Schema Theorem, the motivation for a schema definition is to distinguish partial solutions which will be combined into successively larger partial solutions. We note that S-Expression trees can’t actually be combined without the removal of some subtrees thus fragments must also be partial solutions. Partial solutions will increase in frequency in the population because consistently their sample is above average in fitness and they are unlikely to be disrupted. Multiple fragments (in addition to multiple S-Expression trees) should be considered as schemas, not because all of them fit these criteria, but because some of them do. Fragments will have either equal or greater number of samples in the population than subtrees (when they are less specific). Their disruption likelihood depends upon the relationship between defining length and the size of the programs in which they are embedded so it is not the case that they are always more prone to disruption.

For clarification an S-Expression isomorphically corresponds to a rooted, point-labeled tree with ordered branches in which primitives with no arguments are leaves and the remaining primitives have a child for each of their designated arguments. A fragment isomorphically corresponds to a rooted, point labeled tree of primitives where no primitive without arguments can be an internal node.

**Definition 1:** A *GP-Schema* is a set of ordered pairs: $H = \{(a_1, b_1) \ldots (a_n, b_n)\}$.

The partial function $H: A \rightarrow N$ maps either an S-Expression or fragment to the cardinality of the matching function (i.e. the number of times it must be distinctly found in an individual).

**Definition 2:** The *Matching Function* of a *GP-Schema* $H$ and a genotype $G$: $\phi(H,G)$ assigns to $H$ and $G$ a set of instantiations. An instantiation is a one to one match between the graph corresponding to the schema's fragments and S-Expressions and some number of subgraphs of the genotype's corresponding tree.
The schema is \{( + \ 3 \ 4 \), \ 2\} and it is instantiated once in Tree A and 6 times in Tree B (AB, AC, AD, BC, BD, CD)

Figure 2 GP-Schema Examples

Figure 2 highlights that according to Definition 1 a schema is defined by one or more S-Expression trees or fragments and that these trees or fragments may be duplicates. This clarifies that whether an individual instantiates a GP-Schema depends upon it containing at least the number of occurrences of subtrees and fragments in the GP-Schema definition. The Matching Function implies that no node corresponding to a primitive may be used more than once in an instantiation but the set of instantiations of a schema in a program could possibly refer to the same node more than once. The same program may instantiate a schema more than once because the schema may be contained in the program multiple times. In contrast with a GA, membership is distinct from instantiation.

**Definition 3:** The order of a GP-Schema is the number of nodes in the graphs corresponding to its S-expressions and fragments. GP-Schemas of low order are less specific than those of higher order, other things being equal.

The schema elements are encircled and the notches identify the path connecting them. The variable defining length is 4 and the fixed defining length is 12.

Figure 3 Example of Defining Length

**Definition 4:** The defining length of a schema is the sum of its variable and fixed defining lengths. The fixed defining length is the number of edges within each S-expression or fragment. It is derivable from the GP-Schema definition alone, independent of the individual instantiating it. The variable defining length is the number of edges which separate the group of S-expressions or fragments. It amounts to the length of the shortest path to a common ancestor excluding
edges which are counted in the fixed defining length. This quantity must be calculated for each instantiator of the GP-Schema with reference to the GP-Schema instantiator. See Figure 3 above.

The defining length of a schema is one factor in the likelihood of its disruption. Divided by the size of the GP-schema instantiator, for the GP-Schema instantiator \( h' \), it yields the probability of disruption \( p_d(h') \). Assuming there is no bias in crossover point selection for each instantiator \( h' \) of schema \( H \):

\[
p_d(h') = \frac{D_{L \text{fixed}}(H) + D_{L \text{var}}(h', H)}{\text{Size}(h')}
\]

and since \( D_L(H) = D_{L \text{fixed}}(H) + D_{L \text{var}}(h',H) \)

\[
p_d(h') = \frac{D_L(H)}{\text{Size}(h')}
\]

Most reported experiments of GP are run with a probabilistic bias in the crossover point selection. Leaf crossover points are probabilistically selected with \( L_b = 0.1 \) and interior points are probabilistically selected with bias \( 1 - L_b = 0.9 \). The number of leaf crossover points \( \nu(H) \) is directly obtainable from the schema definition. This factor creates a more complicated but precise formulation of \( p_d(h') \):

\[
p_d(h) = \frac{L_b \nu(H) + (1 - L_b)(D_L(H) - \nu(H))}{\text{Size}(h')}
\]

Compactness \( C(h') \) relates to disruption in the following way: \( C(h') = 1 - p_d(h') \). Thus when the probability of disruption is high, the compactness is low, and, when the probability of disruption is low, compactness is high.

Where \( i(H,t) \) is the number of instantiators of \( H \) at time \( t \), the average probability that a schema will be disrupted \( \bar{p}_d(H) \) is

\[
\bar{p}_d(H) = \frac{\sum_{h}^{i(H,t)} p_d(h)}{i(H,t)}
\]

An average is necessary because each instantiator of the schema has a different probability of disruption. This is because of two variable elements in a schema instantiator \( h' \): its variable defining length \( D_{L \text{var}}(h',H) \) and the size of the program which contains the instantiator, \( \text{Size}(h') \). Note that thus far we decline to define compactness or disruption directly for a schema but it is defined for an instantiator.
Section 2: GP Schema Theorem

Recall that the Schema Theorem estimates the change in membership of a schema in the population from time $t$ to $t+1$. It has three factors: the membership of the schema at time $t$, the reproductive factor of the schema contributed by fitness proportional selection, and the probability that crossover and mutation will disrupt a schema once it has been reproduced. In consideration of the GP crossover operator and the canonical GP's lack of a mutation operator the third factor needs to be revised in a Schema Theorem for GP. The average probability that a GP-Schema $H$ survives crossover $\bar{P}_s(H)$ equals $1 - XO * \bar{P}_d(H)$ where $XO$ is the probability of crossover.

This results in the GP-Schema Theorem where $\bar{P}_d$ is represented by the random variable over time $\bar{D}_t$, $f(H)$ is the observed fitness of $H$ and $\bar{f}$ is the average fitness of the population

**GP Schema Theorem:**

$$i(H,t+1) \geq i(H,t) \frac{f(H)}{\bar{f}} (1 - XO) \bar{D}_t$$

The interesting new aspect in the estimate are the variable elements of defining length and program size within the compactness factor. The size and shape of a program changes each generation and the average size of programs can change. This is dependent upon the sizes of the above average schemas because they are allocated proportionally more trials in the next generation as shown by the fitness proportional selection term in the theorem. It is also dependent upon the size distribution in the initial population. This implies that the term in the Schema Theorem for likelihood of disruption must be considered a random variable. A formulation of the likelihood of disruption which is not a random variable is stymied by the fact that there seems to be no closed form for estimating or calculating how schemas of above average fitness are distributed across programs of different sizes. It is also incorrect to assume that programs of a certain fitness are of a certain size because schemas of above average fitness may be distributed across programs of different sizes.

Interpreting the random variable $\bar{D}_t$ in fact yields the definitions of disruption and compactness for a schema. Recall that previously disruption and compactness were defined only for members of a schema.

**Definition 5:** $\bar{D}_t$ is the average probability of disruption of schema $H$. Let $E$ be the event that $\bar{D}_t$ is less than $\beta$ a constant. Disruption is defined as probability of event $E$, $Pr(E)$. For $Pr(E) < \alpha$ a schema is disruption prone.

**Definition 6:** Compactness is defined as $1 - Pr(E)$. For $Pr(E) > \alpha$ a schema is compact.

The effect of the reproduction and crossover terms is that schema $H$ grows or decays depending upon a multiplication factor. The factor depends upon two things: whether the schema is above or below the population average in fitness and whether the schema is compact.
Compact schemas with above average observed performance (GP Building Blocks) will be sampled at exponentially increasing rates.

Interpreting implicit parallelism and the K-Armed Bandit analogy with GP is more problematic. Assuming an analogy with the K-Armed Bandit problem, a GA solves trial allocation competitions in parallel with each competition corresponding to a separate K-Armed Bandit problem. Schemas within a competition correspond to arms and samples within a competition are allocated by a K-Armed Bandit strategy [Goldberg 1989]. (There are caveats on this, see [Grefenstette and Baker 1989].) The issue when trying to transfer the result to GP is that the interpretation assumes that trial allocation competition takes place among schemas which have common features but where each “competitor” differs in expression of that feature. Together, the GA encoding scheme (based upon features) and the GA schema definition facilitate this. The lack of a feature-expression orientation (i.e. GP is non-homologous) in the GP representation results in an unclear notion of which schemas compete for trial allocation.

The most satisfactory interpretation we have to date is that the competition basis of GP-Schemas is compactness. That is, GP-Schemas competing for trial allocation have the same vulnerability to disruption. Simplistically, within each program of the population there exist some number of subsequences which match with GP-schemas such that the probability of disruption for those schemas matching the subsequences is the same. A program \( P \) contains some number \( X_p \) of schemas with probability of disruption \( d \). Thus, in a population of size \( N \) at time \( t \) the number of trials to be allocated amongst schemas with probability of disruption \( d \) is:

\[
\sum_{p=1}^{N} x_p \cdot d
\]

This number varies each generation because the size and shapes of programs change in GP due to selection and crossover. Thus in a GA the amount of trials allocated never changes but in GP it does. The fact that the number of competitive trials is non-stationary bears a similarity with the Schema Theorem itself. The Schema Theorem describes behavior for one generation step only. Subsequent to each generation the Schema Theorem holds only given the new distribution of schema fitness. That is why the Schema Theorem must be carefully interpreted in terms of schema with low variance fitness (and compactness in the case of GP).

Perhaps this situation indicates that schema processing may not be the best abstraction with which to analyze GP. The analysis needs a schema definition in which the same group of schemas always competes for trial allocation. But in GP, due to the non-homologous representation, such a static definition is impossible. Our interpretation also implies the same schema may simultaneously be in more than one competition (because in different contexts its compactness will vary) and that competing GP-schemas must also coadapt because they must coexist within the same program. This seems to make schema-based analysis intractable.
Section 3: GP Building Blocks and a GP Building Block Hypothesis

Our careful formulation of the GP Schema Theorem and Definitions 7 and 8 below support a more detailed transfer of GA theory to GP than was heretofore justified, and enable us to construct a testable hypothetical account of how GP searches by combining building blocks. While gaps in existing GA theory and additional difficulties intrinsic to GP - especially its variable-length genotype - prevent the resulting account from being a full fledged theory of how GP works, it is at least detailed and explicit enough to suggest a rich set of computational experiments and to admit of confirmation or disconfirmation [O’Reilly and Oppacher 1994].

Definition 7: A building block in GP is a compact schema (see definitions 5 and 6) with above average observed performance.

Definition 8: The GP Building Block Hypothesis is that GP combines building blocks, the best partial solutions of past samplings, to compose individuals which are of improving fitness. The source of GP’s power, (i.e. when it works), is that it follows this heuristic of combinational construction rather than trying every conceivable combination.

Experimentation with explicit schema based functions (so called potential “Royal Roads” for GAs) has identified overlapping stages of search and the fact that the role of the crossover operator varies considerably throughout the course of the GA search [Forrest and Mitchell 1992]. To flesh out our GP Building Block Hypothesis we assume this framework and try to resolve how each stage would take place in GP.

Stage 1. Discovery of Primary Building Blocks

During Stage 1, above average fit schemas are discovered as schemas compete with respect to compactness to dominate possible positions in the programs of the population. Since competing schemas may have to coexist in programs, it is undesirable that one schema should achieve early domination in the population. Such above average, diverse schemas that do not yet dominate in numbers are primary building blocks.

Stage 1 is characterized by the fact that the average compactness of schemas grows with the average size of programs. Programs grow in size simply because population initialization uses a smaller maximum height than the maximum height constraining the crossover operator. This assured growth may permit the ongoing discovery of progressively larger primary building blocks. It should be noted that the role of crossover is strictly exploratory during this stage: the discovery of primary building blocks proceeds independently of the crossover operator recombining building blocks to form blocks of equal size. It may be the case that some GP runs converge during Stage 1 alone, i.e., their convergence is due entirely to program growth and not at all to crossover.

The overall increase in program size might also provide a "timing cushion" which allows primary building blocks of the same compactness to coevolve and increase their numbers despite being discovered at different times. The growth in size obviously makes room for new building
blocks because a program's capacity to accommodate new subtrees is directly related to its size. It is plausible that growth and the rate of increase of existing building blocks have opportunistic periods when narrow dominance is subsumed by the size increase. See Figure 4 for a graphical view of the concept.

The solid line denotes the quantity of building blocks of a given compactness which exist in the population at time t when a new building block is found. The striped vertical line indicates the time point when size exactly accommodates all building blocks. Prior to that a new building block not only has the opportunity to compete for slots occupied by existing blocks but it can use the newly available slots arising through growth.

The Schema Theorem cannot describe the exact nature of the rise in primary building block quantities. This scenario is complicated: a diverse set of schemas arises amidst program growth, building block discovery at different generations, unequal (though above average) fitness of schemas and the bias of selection.

**Stage 2: Building Block Combination**

Stage 1 forms a diverse set of building blocks which are strongly represented in the population. These building blocks are recombined during Stage 2 to form successively larger building blocks. Stage 2 is characterized by the primary role of crossover in exploitative recombination. But by changing the probability of schema disruption, the change in average program size also influences the ability of crossover to effectively combine building blocks.

Are successive layers of building blocks larger and larger subtrees? That is, is GP search hierarchical because its crossover operator develops bigger and better partial solutions strictly by tree expansion? It seems clear that layers composed of building blocks which are single trees of the same order are discernible because such blocks are the most compact of the hyperplane defined by the order (assuming that each schema is distributed in programs of the same sizes as the others). But there may also be intermediate layers of building blocks between these 'tree layers' which have blocks that are not single subtrees. The likelihood of finding blocks consisting of several S-expressions is much greater than finding ones which consist of a single S-expression (there are more ways the former schemas could arise) and if they are not significantly less com-
pact a new sublayer could arise. These tree schemas consisting of several S-expressions are un-
ordered collections of good pieces. The importance of such intermediate 'pieces layers' depends
on whether GP crossover has the power to explore more and different combinations of the pieces
and thus increase the likelihood of detecting the building blocks for the next 'tree layer'. The ef-
effect of crossover in this respect is not observable in one program but across the population where
an exchange and rearrangement of schemas takes place.

**Stage 3: Dominance of High Order Compact Building Blocks**

Stage 3 seems likely to take place once the size of programs becomes relatively stable. Diversity will be maintained in the population because crossover will change the positions building blocks can assume in programs. If the building blocks have been consistently above average in fitness the average population fitness should rise.

**Section 4: Discussion and Future Avenues of Investigation**

To accept the conjecture that a building block process takes place in GP search requires assum-
ing that building blocks do exist. This seems doubtful to us because of the highly epistatic nature of programs and their primitives. The value of a schema depends upon the context in which it occurs. With many different programs and possible settings for a schema, the variance of observed schema fitness will be high. Furthermore, the process requires that building blocks have low disruption likelihood. This circumstance is not assured because GP only indirectly and loosely controls how large program trees become. The size and shape of a program changes each generation in an unpredictable manner. GP crossover, by enforcing an upper bound on height, influences program size and shape in an insensitive, circuitous way. GP may always elude more detailed prediction of its behavior because the probability of schema disruption is a random vari-

able.

But GP does experience success. One explanation is that on these occasions the primitive set and the problem may have properties which induce building blocks. Here it is impossible to sidestep the issue that the success of GP on a particular problem strongly depends upon the choice of primitives and the design of the test suite and fitness function (see [O'Reilly and Oppacher 1992]). The clearest way to confirm that building block processing does take place is to design explicit building block functions. In this case, the existence of building blocks is undeniable and their distribution could be monitored to see if the GP search does indeed progress through the three hypothesized stages. With such potential "Royal Road" functions the role of the GP crossover operator could be studied in more detail and the extent to which building blocks are involved in GP's success could be conclusively determined.

Another explanation which has been hinted at already is that the discovery of successful programs is a fortuitous circumstance mainly due to the serendipitous timing of genotype growth.
and search space exploration. There may be no building blocks at all. If building blocks don’t exist, what is the value of crossover? In Stage 1, crossover only explores. This exploration could be accomplished with a mutation operator. Perhaps crossover is not more effective for hierarchical program search than other search techniques which accommodate GP’s hierarchical variable length representation. Thus the question arises whether a mutation operator for trees (which can shrink and grow them or “flip” the symbols) would be just as effective when coupled with simulated annealing or hill climbing?

Which explanation is appropriate may become clearer if more is known about the nature of fitness landscapes of problems. It would be valuable to characterize the classes of landscapes over which GP is more effective than other search techniques. It seems essential to acquire useful statistics as suggested in [Manderick, de Weger et al. 1991]. We have begun examining fitness correlation coefficients and autocorrelation lengths for GP problems.

**Conclusion:** After having provided a more rigorous formulation of the GP Schema Theorem and Building Block Hypothesis than previously available, we have questioned whether the existence of building blocks always explains the successes of GP. We have also questioned whether building blocks even exist in GP. We continue to search for better explanations as to when and why GP is indeed a superior program search technique.

**Acknowledgments:** The authors wish to thank S. Forrest, T. Jones, B. Macready, M. Mitchell, M. Wineberg, and the members of the UNM study group for their insightful contributions. U.M.O.R. greatly appreciates the stimulating environment provided by the Santa Fe Institute.

**References**


